Journal of KIIT. Vol. 23, No. 6, pp. 181-189, Jun. 30, 2025. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2025.23.6.181

A Tabletop Game Design That Scaffolds Debugging for Programming Beginners

Avila Torres Eric Eduardo*, Shin Chun Kang**

Abstract

Digital games with educational purposes and corresponding user experience measurement via game analytics have been extensively covered in the literature; however, non-digital games such as tabletop games in education and their educational impact as scaffolding techniques have limited research. This paper aims to share the design of such a game, applied to teaching debugging, and to select testing methods to assess its engagement and educational effect on how programming beginners approach debugging. This tabletop game was developed based on design principles from developers, scholars, and other experts who have studied the core elements that make a game playable, engaging, and educational. As a result, by combining game mechanics with narrative immersion, the game successfully achieves both educational objectives and enjoyment, demonstrating an improvement in learning outcomes with over a 30% increase in engagement compared to traditional debugging instruction.

요 약

디지털 게임을 교육적 목적으로 활용하고, 이에 대한 사용자 경험을 게임 분석을 통해 측정하는 연구는 문헌에서 광범위하게 다루어져 왔다. 그러나 교육에서 사용되는 테이블탑 게임과 같은 비디지털 게임 및 이들이 스캐폴딩 기법으로서 가지는 교육적 영향에 대한 연구는 제한적이었다. 본 논문은 디버깅을 가르치는 데 적용된 이러한 유형의 게임 디자인을 공유하고, 프로그래밍 초보자들이 디버깅을 접근하는 방식에 대한 몰입도와 교육적 효과를 평가하기 위한 테스트 방법을 선정하는 것을 목표로 한다. 이 테이블탑 게임은 개발자, 학자, 그리고 게임의 플레이성, 몰입성, 교육적 요소를 연구해 온 전문가들의 디자인 원칙을 바탕으로 개발되었다. 결과적으로 게임 메커니즘과 서사적 몰입도를 결합하여 교육적 목표와 재미를 충족시킬 수 있으며, 전통적인 디버깅 학습 방식보다 평균 30% 이상의 높은 몰입도 향상을 통한 학습 효과 향상을 보여줌을 확인하였다.

Keywords
unplugged computational thinking, game-based learning, game design, debugging education

* Dept. of Computer Education and College of Education, Kongju National University
- ORCID: https://orcid.org/0009-0000-6652-3048
** Dept. of Computer Education and College of Education, Kongju National University
- ORCID: https://orcid.org/0009-0009-3121-4347

· Received: Sep. 12, 2024, Revised: Apr. 16, 2025, Accepted: Apr. 19, 2025
· Corresponding Author: Shin Chun Kang
Department of Computer Education, Kongju National University, Gongju-si, 32588, South Korea
Tel.: +82-41-850-8824, Email: godsky@naver.com

Ⅰ. Introduction

Jeannette M. Wing's 2006 essay "Computational Thinking" introduced the idea that Computational Thinking (CT) is a fundamental skill for the 21st century[1]. A critical component of CT, especially in programming education, is debugging-the process of detecting, analyzing, and correcting errors in code. Debugging is essential for problem-solving and developing logical thinking, but it is often one of the most challenging skills for programming beginners due to the cognitive load and systematic thinking required. Research supports that game-based learning offers an effective platform for teaching debugging and other IT skills, providing students with engaging, risk-free environments to explore complex concepts like debugging[2]. These environments allow for iterative problem-solving and promote resilience, both critical for effective debugging.

Debugging education often involves structured learning activities focusing on skills such as error identification, hypothesis testing, and iterative correction[3]. Game-based and unplugged learning activities provide accessible ways to scaffold these skills, showing that structured game mechanics can effectively teach systematic approaches to debugging[4].

This paper aims to present an unplugged tabletop game, HEXA_766, designed to teach and scaffold debugging for programming beginners. This game focuses on developing a meta-cognitive approach to debugging through an engaging game design. We will explore the game's design principles, prototyping, and testing methods, with an emphasis on both user engagement and educational impact.

Ⅱ. Related Works

CT is widely recognized as a critical skill for students in the 21st century. Research on CT education often emphasizes the use of game-based learning to engage students in problem-solving and algorithmic thinking. However, there has been limited research directly focusing on scaffolding the debugging process through tabletop games, particularly unplugged learning environments.

2.1 Debugging in educational tools

The game we propose has an educational purpose which entails it falls within the type of games classified as serious. According to [5], "a serious game is one with elements that besides promoting fun also have a special purpose." Additionally, [6] stated that the growing relevance of computational thinking has led to an increased demand for computing education in educational facilities while lowering the costs required to build infrastructure". Likewise, unplugged educational resources provide computing education in educational facilities while lowering the investment on hardware and software.

Previous studies have extensively explored debugging from an educational perspective. [3] emphasize that novice programmers often approach debugging inefficiently, relying on trial and error. They call for more structured methods to teach debugging systematically, which aligns with our approach in HEXA_766, where players must debug their actions using a fictional programming language. The game encourages players to identify and correct errors in their commands, guided by a Game Master who helps scaffold the debugging process.

It was proposed that structured debugging activities are essential for IT education, noting that step-by-step problem-solving activities are key to fostering students' confidence and skill in debugging[7]. Likewise, they argue that structured debugging activities are essential for teaching effective debugging skills. While their research focuses on digital tools, the structured, step-by-step debugging process in HEXA_766 provides a non-digital alternative. The Game Master serves a

role akin to automated feedback in digital environments, guiding players as they reflect on and correct their errors.

2.2 Tabletop Games and Debugging

Tabletop games have been increasingly recognized for their ability to teach computational thinking and problem-solving skills. Some existing tabletop games already incorporate debugging concepts, though indirectly. Unplugged games and activities have shown promise in teaching computational thinking and debugging. The CS Unplugged was introduced to allow students to practice CT concepts without computers, showing that hands-on learning can be highly effective in building foundational IT skills[4].

It was found that unplugged activities are effective for younger students learning computational concepts by reducing cognitive load and enhancing focus on logical problem-solving[8]. HEXA_766 extends this approach, using an unplugged, tabletop game format to specifically target debugging.

Although research on games specifically designed for debugging is limited, a few tabletop games touch on computational thinking, logical reasoning, or problem-solving in a way that relates to debugging. Here are some notable examples. RoboRally encourages players to program their robots' movements across a grid. Players often encounter situations where their robots don't behave as expected, prompting them to adjust their plans mid-game-an activity that mirrors debugging. Like HEXA_766, players learn through iterative problem-solving and reflection on their errors.

Code Master provides players with a set of instructions to guide an avatar through logic puzzles. Players must debug their initial plans when they encounter obstacles, fostering logical thinking and problem-solving skills akin to those required for debugging. Similarly, in HEXA_766, players program their robotic units and must adjust their commands when things go wrong.

Robot Turtles introduces young players to programming by having them sequence actions using cards. If the turtle doesn't reach the goal, players must "debug" their commands by rethinking their sequence. This process aligns with our design in HEXA_766, where players must debug their code to avoid errors and progress in the game.

Turing Tumble is a mechanical board game where players create logic circuits to solve puzzles. Debugging is a core part of the gameplay, as players frequently need to adjust their circuits when they don't behave as expected. Similarly, HEXA_766 scaffolds the debugging process by encouraging players to think systematically about their mistakes and refine their strategies.

These games highlight the growing trend of using tabletop games to teach computational concepts, but none explicitly focus on debugging as a primary educational goal. HEXA_766 fills this gap by directly targeting debugging skills through its card-based programming system and the role of the Game Master, which is designed to mimic real-world debugging environments in a collaborative, non-digital setting.

2.3 Comparison of previous works

While the use of digital games in education has been extensively studied, fewer works address the unplugged, tabletop format for teaching debugging. [9] and [2] have demonstrated that digital games can improve student engagement and learning outcomes in programming, but their focus is primarily on algorithmic thinking and general problem-solving, with limited emphasis on debugging. In contrast, HEXA_766 integrates the debugging process as a central mechanic, providing immediate and structured opportunities for players to correct errors in their commands. Unlike digital games that automate feedback, HEXA_766 incorporates a Game Master who actively engages players in the debugging process, encouraging critical thinking and reflection.

III. Game Design

Figure 1 shows the whole concept of the HEXA_766 game. The design of HEXA_766 is informed by game design principles that align with information technology and programming education. Our aim was to satisfy conflicting constraints to arrive to an artifact of systematic coherence[10], emphasizing elements of computational thinking and debugging and showing how game mechanics can support IT learning objectives[11]. Debugging, a systematic approach to identifying and correcting errors, is critical for programming and requires skills such as logical reasoning, attention to detail, and problem-solving. HEXA-766 addresses these needs through structured gameplay elements that directly support IT learning objectives.

In this section, we describe the tenets and core components of the game design and how each contributes to the game's educational and engagement goals.

3.1 Tenets of game design for IT learning

For the Unholy Alliance, by immersing players in problem-solving scenarios, this principle engages players in debugging tasks that require iterative hypothesis testing and correction, supporting systematic IT thinking[12]. For the MAD Principle, consistent game mechanics reinforce logical patterns, mirroring the consistency needed in real-world debugging, where identifying dependencies is critical to resolving errors[13].

Using the Player's Imagination, this principle allows for creative solutions, fostering IT skills like logical sequencing and troubleshooting, where players use minimalistic commands to solve complex problems, an approach supported by studies on computational thinking[14].

3.2 Game design elements for debugging education

Each component of HEXA-766 is designed to foster IT skills, particularly focusing on debugging. These elements work together to simulate real-world debugging processes in a non-digital, hands-on format. Fictional Programming Language Cards: The game introduces a fictional programming language through a set of command cards that players use to control robotic units. Players must arrange these cards logically to achieve desired outcomes on the game board. This activity simulates algorithmic thinking and provides a direct analogy to writing code, where players must debug sequences by identifying and correcting logical errors in their command arrangements.

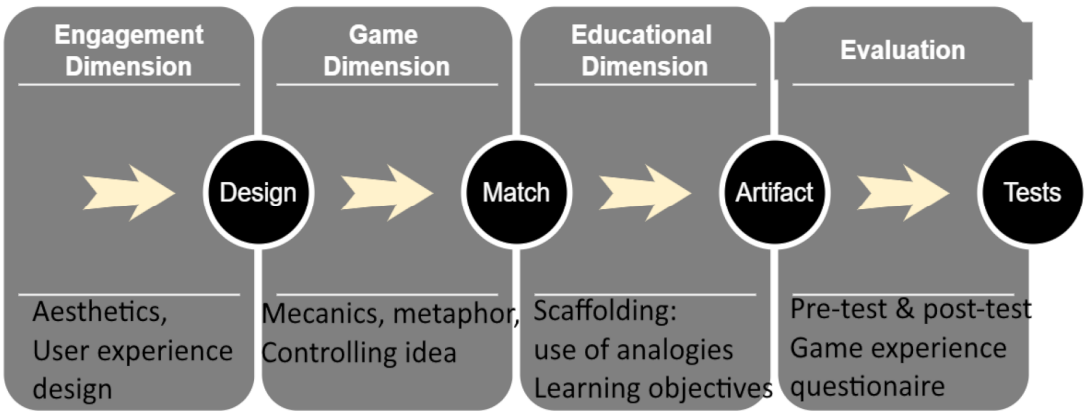

Fig. 1. HEXA_766 game design concept

The Game Master in HEXA-766 guides players by providing structured feedback when commands fail. This mirrors the real-world role of a debugging environment in an Integrated Development Environment (IDE) or a programming instructor who supports novices. By receiving hints or partial solutions from the Game Master, players are encouraged to analyze their errors and refine their approach, mimicking iterative debugging in IT.

HEXA-766 requires players to repeatedly test, adjust, and re-test their commands. Each round of gameplay represents an iteration, where players encounter obstacles, troubleshoot their errors, and refine their strategies. This iterative approach is fundamental to debugging in programming, as players learn to break down complex problems, identify specific errors, and adapt their solutions systematically.

In order to complete tasks on the board successfully, players must sequence commands logically. When their actions do not lead to the intended result, players are prompted to identify specific points of failure. This activity directly supports the development of error identification skills, which are essential in IT for understanding and correcting coding mistakes.

3.3 IT skills development through game

Through these design elements, HEXA-766 teaches key debugging concepts and supports foundational IT skills, including the followings.

Error Correction, where players practice diagnosing and correcting errors, a core aspect of IT programming tasks. Logical and Algorithmic Thinking, where by structuring commands in sequences, players engage in algorithmic thinking, similar to programming workflows in IT. Finally for the Problem-Solving Resilience, iterative testing and feedback from the Game Master foster a resilience to failure, which is essential for successful debugging and IT troubleshooting.

3.4 Engagement dimension

The game is composed of a modular board made of hexagonal pieces that represent an extraterrestrial environment for players to explore. Each player has a set of four pieces representing robotic units they command during gameplay. The game includes a set of heart-shaped tokens representing an energy system and a deck of cards used to "program" the attack-defense system of the units. Additional sticks and rings indicate unit status, while dice establish movement mechanics. The dice mechanic in HEXA-766 introduces an element of chance, impacting how players control their robotic units as they navigate the board. Although this mechanic does not directly simulate programming concepts, it serves a critical role in maintaining player engagement, an essential factor for sustained educational impact. The game components, except for the card deck, were 3D printed to facilitate ease of play and user engagement.

3.5 Game dimension

HEXA_766 employs a narrative-driven "controlling idea" [15] to immerse players in the role of space archaeologists. commanding robotic units. Players compete to explore a hostile alien environment and retrieve valuable alien technology by programming scavenging drones using a fictional language.

3.6 Mechanics

Figure 2 shows the key components of HEXA_766. In HEXA_766, players move their robotic units based on dice rolls, deciding how to allocate movement points across units. Players must navigate areas of varying damage levels and retrieve "sticks" representing missing parts of an AI system. Each stick placed in the "A.I. Station" earns points, symbolizing the reactivation of the alien intelligence. Players also

access a programming deck to create conditions for defending or attacking units, a central feature of the educational dimension, as it requires debugging of statements within the game context.

In HEXA-766, players use a deck of fictional programming language cards to create command sequences for their robotic units. The card-based mechanics align with principles of modular game design [15] and are the "core" of the game's educational experience, simulating basic programming tasks that support the development of debugging skills. Each card represents a specific action or command that players must combine logically to achieve in-game objectives, akin to writing code in a programming language.

The deck contains several types of cards, each representing different entities, actions, or logics that mimic programming structures. For Unit Cards, these cards allow the player to refer to specific robots or groups of robots on the battlefield. As for Attack and Defense Cards, these cards specify actions the robot can take to attack enemy units or to defend from the opponent's tactics, representing conditional actions. Players must decide when to use these cards based on in-game scenarios, simulating decision-making in if-else statements. They serve as basic "commands," like function calls in programming.

As for the Structure cards, they include arithmetic, assignment, comparison / logical operators, and control flow statements. Just like in the popular game and anime Yu-!Gi!-Oh!, Players draw cards from the deck and arrange them to program actions that can take place in each of the turns of the game. For example, a player might use a sequence such as “If scout >= 40 : Attack All Opponents”. To correct the command, the player could rewrite it as “Debugged Condition: If scout >= 40 Attack All Opponents == -20 points”.

Here, the player specifies an attack value of 20 points, ensuring that the condition fully defines the action. This adjustment requires players to carefully consider logical completeness and clarity, similar to debugging code to prevent errors or undefined behaviors. To create this specific condition, the player would need to gather 6 cards and a plastic piece with the colon symbol (:).

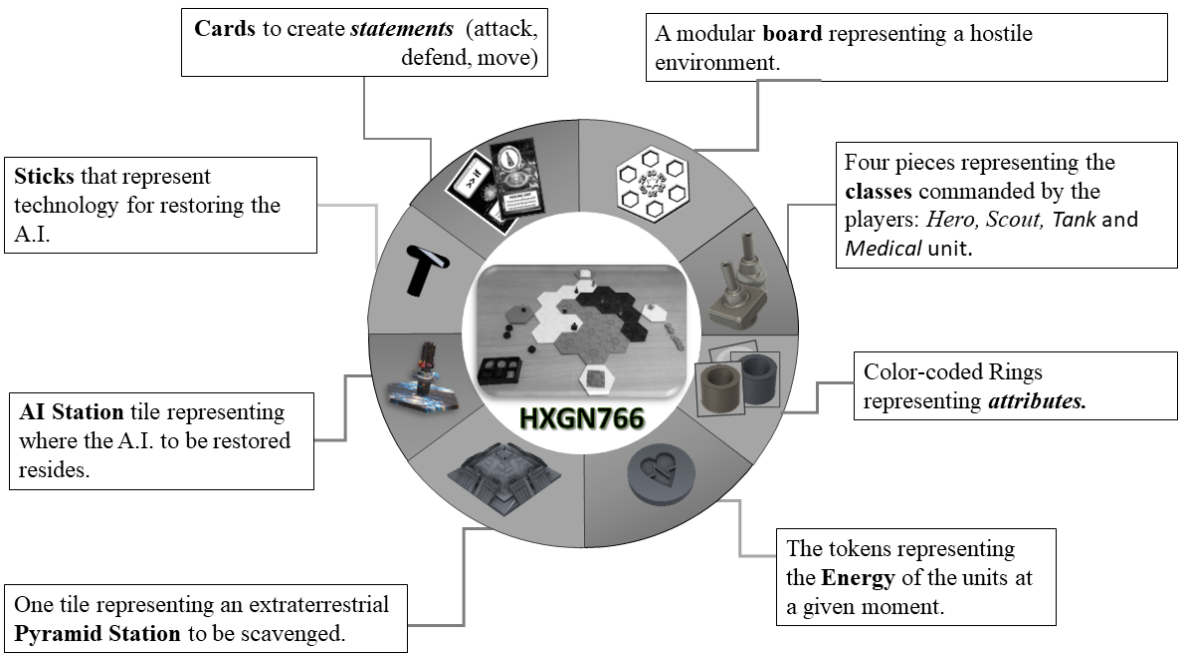

Fig. 2. Components of HEXA_766

3.7 Educational dimension

Debugging is a fundamental skill in programming that requires more than simply identifying mistakes; it requires players to think systematically and iteratively. The design of HEXA_766 addresses these needs by scaffolding debugging practices directly into the game mechanics. The game uses a fictional programming language embedded in a deck of cards, which allows players to create commands for their robotic units. Players are then prompted to identify and correct errors in their attack, defend and move commands as they navigate the board. Additionally, the Game Master provides structured guidance when players' programming conditions fail, encouraging systematic thinking and reflection on errors, similar to debugging processes in actual programming.

HEXA_766 incorporates debugging education by creating situations where players encounter obstacles due to incorrect or incomplete programming commands. To progress, they must debug these commands, examining their logic and adjusting it to achieve desired results. The Game Master plays a crucial role by offering hints and structured feedback when a command does not work as intended, mirroring the real-world debugging process in programming. This setup teaches players (1) to identify errors: players learn to spot mistakes in their commands, understanding which actions cause unintended results, (2) to analyze and correct that they analyze the root cause of errors, revising their strategies and commands accordingly, and (3) to reflect and test. By iterating through their commands, players experience a trial-and-error learning approach, solidifying debugging concepts.

These elements of HEXA_766 engage players in an authentic debugging experience, where they practice debugging in a controlled, supportive environment.

IV. Evaluation and Experimental Results

To empirically validate the educational efficacy of HEXA_766, we designed a controlled experiment comparing its impact on debugging skills against traditional instructional methods. The study involved 100 novice programmers aged 12-17, evenly divided into an experimental group (using HEXA_766) and a control group (using worksheet-based tasks). Both groups had comparable baseline programming experience ($\leq$3 months).

The experiment addressed two research questions: RQ1, whether HEXA_766 improves debugging skills more effectively than traditional methods, and RQ2, how engagement with the game correlates with learning outcomes. Participants completed a 30-minute pre-test assessing their ability to identify logical errors in conditional statements and debug faulty code snippets. The experimental group then engaged with HEXA_766 for two hours, guided by a structured protocol that included gameplay and reflection sessions. The control group solved equivalent debugging problems through worksheets with instructor feedback.

Quantitative results demonstrated significant differences between groups. The experimental group improved debugging accuracy by 30% (pre-test: 52%, post-test: 82%), while the control group improved by 18% (pre-test: 50%, post-test: 68%). A two-tailed t-test confirmed statistical significance ($t(98)=3.21$, $p<0.01$). Additionally, the experimental group reduced time spent diagnosing errors by 40%, suggesting enhanced efficiency in problem-solving.

Engagement metrics, measured via the Game Experience Questionnaire (GEQ), revealed high levels of enjoyment (4.5/5) and flow (4.2/5) among HEXA_766 players. Qualitative analysis of post-experiment interviews highlighted recurring themes: 85% of participants noted that iterative gameplay helped them "view errors as opportunities to learn," while 78% emphasized the Game Master's scaffolding as “critical for understanding why commands failed”.

Comparative analysis further underscored HEXA_766's advantages. The experimental group outperformed the control group in correcting logical errors ($p<0.05$) and articulating step-by-step debugging strategies ($p<0.01$). A moderate positive correlation ($r=0.72$) between GEQ engagement scores and post-test performance suggests that enjoyment and immersion directly contribute to learning gains.

While these results strongly support HEXA_766's efficacy, we acknowledge limitations. The sample size (N=100), though sufficient for initial validation, warrants expansion in future studies. Additionally, the short-term assessment window precludes conclusions about long-term skill retention. To address this, we plan longitudinal tracking of participants' debugging performance over six months.

Ⅴ. Conclusions

In this paper, we presented HEXA_766, a tabletop game meticulously designed to scaffold the process of debugging for beginners in programming. By drawing from established concepts of game design, educational theory, and cognitive psychology, we crafted an immersive and engaging experience that introduces players to fundamental programming concepts while captivating them within a sci-fi narrative.

Our approach was rooted in the idea that games possess the unique potential to make learning abstract and complex concepts more accessible. HEXA_766 embodies this philosophy by combining educational objectives with captivating game mechanics, weaving a narrative that transports players into the role of space archaeologists commanding robotic units in a hostile extraterrestrial environment. By implementing analogies and a fictional programming language, HEXA_766 not only entertains but also enhances players' meta-cognition of computational thinking elements, especially in relation to the debugging process.

In our evaluation, we apply a multi-method approach to assess the user experience and educational impact of HEXA_766. By employing tools such as the GEQ, and narrative interviews, we aimed to gain comprehensive insights into players' engagement and learning outcomes. Future work includes testing engagement factors through a blended digital-analog version of the game.

References

[1] J. M. Wing, "Computational thinking", Commun. ACM, Vol. 49, No. 3, pp. 33-35, Mar. 2006. https://doi.org/10.1145/1118178.111821.

[2] M. Papastergiou, "Digital game-based learning in high school computer science education", Comput. Educ., Vol. 52, No. 1, pp. 1-12, Jan. 2009. https://doi.org/10.1016/j.compedu.2008.06.004.

[3] R. McCauley, et al., "Debugging: A review of the literature from an educational perspective", Comput. Sci. Educ., Vol. 18, No. 2, pp. 67-92, Jun. 2008. https://doi.org/10.1080/08993400802114581.

[4] T. Bell, et al., "Computer science unplugged: School students doing real computing without computers", New Zealand J. Appl. Comput. Inf. Technol., Vol. 13, No. 1, pp. 20-29, Mar. 2009.

[5] J. Ahn and S.-J. Park, "The effects of serious game for elderly people: Focused on the case of development of 'Rejuvenesce Village'", J. KIIT, Vol. 14, No. 3, pp. 195-206, Mar. 2016. https://doi.org/10.14801/jkiit.2016.14.3.195.

[6] Y. Jung and N. Park, "Design of unplugged learning tools for cyber attack and defense hacking principle", J. KIIT, Vol. 19, No. 5, pp. 111-119, May 2021. https://doi.org/10.14801/jkiit.2021.19.5.111.

[7] L. Chen, et al., "Towards a framework for teaching debugging", in Proc. 21st Australas. Comput. Educ. Conf., Sydney NSW Australia, pp. 1-10, Jan. 2019. https://doi.org/10.1145/3286960.3286961.

[8] C. P. Brackmann, et al., "Development of computational thinking skills through unplugged activities in primary school", Proc. 12th Workshop Primary Secondary Comput. Educ., Nijmegen Netherlands, pp. 65-72, Nov. 2017. https://doi.org/10.1145/3137065.3137069.

[9] M. Prensky, Digital Game-Based Learning. New York, NY, USA: McGraw-Hill, 2001.

[10] R. Hunicke, M. LeBlanc, and R. Zubek, "MDA: A formal approach to game design and game research", Proc. AAAI Workshop Challenges Game AI, Vol. 4, No. 1, pp. 1722, pp. 1-5, Jul. 2004.

[11] J. P. Gee, What Video Games Have to Teach Us About Learning and Literacy. New York, NY, USA: Palgrave Macmillan, 2003.

[12] S. Meier, "The psychology of game design", GDC Vault, 2010. https://gdcvault.com/play.

[13] Adams and J. Dormans, Game Mechanics: Advanced Game Design. Berkeley, CA, USA: New Riders, 2012.

[14] J. Shute, M. Ventura, and F. Ke, "The power of play: The effects of Portal 2 and Lumosity on cognitive skills", Comput. Educ., Vol. 80, pp. 58-67, Jan. 2015. https://doi.org/10.1016/j.compedu.2014.08.013.

[15] K. Ahlquist, "Game design philosophy: The controlling idea", DR Wictz Blog, Apr. 2014. https://dr.wictz.com/2014/04/game-design-philosophy-controlling-idea.html.

Authors

Avila Torres Eric Eduardo

1999. 3 : BS degree, Dept. of Philology and Languages, National University of Colombia

2017. 10 : MS degree, Dept. of Design and Management of Technological Projects, International University of La Rioja

2018. 3 ~ present : Ph.D candidate, Dept. of Computer Education, Kongju National University

Research interests : STEM, Serious Games, Natural Language Processing

Shin Chun Kang

1993. 2 : BS degree, Dept. of Education, Busan National University

1999. 2 : MS degree, Dept. of Curriculum Study, Korea National University of Education

2003. 3 : Ph.D degree, Dept. of Educational Technology, Korea National University of Education

2005. 3 ~ present : Professor, Dept. of Computer Education and College of Education, Kongju National University

Research interests : Computer Education, Educational Technology, Artificial Intelligence Convergence Education